User Modeling in Adaptive Hypermedia Applications

Hongjing Wu, Geert-Jan Houben[1], Paul De Bra

Department of Computing Science
Eindhoven University of Technology
PO Box 513, 5600 MB Eindhoven
the Netherlands

phone: +31 40 2472733
fax: +31 40 2463992
email: {hongjing,houben,debra}@win.tue.nl

Abstract: A hypermedia application offers its users a lot of freedom to navigate through a large hyperspace. The rich link structure of the hypermedia application can not only cause users to get lost in the hyperspace, but can also lead to comprehension problems because different users may be interested in different pieces of information or a different level of detail or difficulty. Adaptive hypermedia systems (or AHS for short) aim at overcoming these problems by providing adaptive navigation support and adaptive content. The adaptation is based on a *user model* that represents relevant aspects about the user. The adaptive navigation support and the adaptive content will reduce the orientation problems and the comprehension problems respectively.

The AHS AHA [DC98], that we developed at the Eindhoven University of Technology, and its predecessors [CD97, DC97], did only support user knowledge at the page level and not on the level of large abstract concepts, which limits adaptation to a low knowledge level. Other current AHS appear to suffer from similar problems. As part of the redesign process for AHA we have developed a reference model for the architecture of adaptive hypermedia applications, named AHAM (for Adaptive Hypermedia Application Model) [DHW99], which is an extension of the Dexter hypermedia reference model [HS90, HS94]. In AHAM knowledge is represented through hierarchies of large composite abstract concepts as well as small atomic ones. AHAM also divides the different aspects of an AHS into a *domain model* (DM), a *user model* (UM) and a *teaching model* (TM). This division provides a clear separation of concerns when developing an adaptive hypermedia application.

In this paper, we specifically concentrate on the user modeling aspects of AHAM, but also describe how they relate to the domain model and the teaching model. First, we introduce general concepts involved in adaptive hypermedia applications and their design and use the AHAM reference model to describe the different aspects, specially domain model, user model and teaching model. We show how user features are modeled and how the actual adaptation process is performed. We illustrate this general approach for user modeling and adaptation by considering the AHS AHA.

Keywords: adaptive hypermedia, user modeling, adaptive presentation, adaptive navigation, hypermedia reference model

---

[1] On leave at the Department of Mathematics and Computer Science of the University of Antwerp (UIA), Belgium.

1. Introduction

Hypermedia systems in general and Web-based systems in particular are becoming increasingly popular as tools for user-driven access to information. One of the characteristic properties of hypermedia applications is that they offer users a lot of freedom to navigate through a large hyperspace. By choosing between hyperlinks users can follow very different paths through the hyperspace in order to access the contents contained in that hyperspace. Unfortunately, this rich link structure of the hypermedia application causes some serious usability problems:

- A typical hypermedia system always presents the same links on a page, regardless of the path a user followed to reach this page. When the system wants to providing navigational help, it does not know which part of the link structure is most important for the user. So, if for example it wants to provide a map of the structure, it does not know exactly what is relevant for this particular user: the map cannot be simplified by filtering (or graying) out links that are less relevant for the user. Not having personalized maps is a typical *navigation problem* of hypermedia applications.
- Navigation in ways that the author did not anticipate also causes *comprehension problems*. For every page the author makes an assumption about the user's foreknowledge. However, there are too many ways to reach a page to make it possible for an author to anticipate all possible variations in foreknowledge when a user visits that page. Therefore, a page is always presented in the same way. This often results in users visiting pages containing a lot of redundant information for them ("too much information") and pages that they cannot fully understand because they lack some expected foreknowledge ("too little information or wrong information").

Adaptive hypermedia systems (or AHS for short) aim at overcoming these problems by providing adaptive navigation support and adaptive content. Adaptive hypermedia is a recent area of research on the crossroad of hypermedia and the area of user-adaptive systems, with applications in educational applications, on-line information systems, on-line help systems, information retrieval systems, etc. The goal of this research is to improve the usability of hypermedia systems by making them personalized. The personalization or adaptation is based on a *user model* that represents relevant aspects about the user. The system gathers information about the user by observing the use of the application, and in particular by observing the *browsing* behavior of the user. An overview of systems, methods and techniques for adaptive hypermedia can be found in [B96].

AHA [DC98] is an AHS system developed out of Web-based courseware for an introductory course on hypermedia at Eindhoven University of Technology. In AHA knowledge is considered at the same level of abstraction as the contents: every concept is represented by a page and knowledge represents whether or not a user knows the concept associated with a page. The user's knowledge about a given concept is a binary value: *known* or *not known*.

AHAM (for Adaptive Hypermedia Application Model) [DHW99] is a reference model for the architecture of adaptive hypermedia applications. It is an extension of the Dexter hypermedia reference model [HS90, HS94], and evolved partially from a redesign process of AHA. The central issue in AHAM is the fact that performing both "useful" and "usable" adaptation in a given application depends on three factors:

- There must be a *domain model* that describes how the information content is structured in terms of relationships between high level and low level concepts, and that indicates how these abstract concepts are tied to pages (that can actually be presented).
- By observing the user's behavior a fine-grained *user model* must be maintained to represent the user's preferences, knowledge (related to concepts from the domain model),

goals, navigation history and possibly other relevant aspects.
- The adaptation of the presentation of both content and link structure to the user's preferences and knowledge level is performed by the system based on some "intelligence". In most AHS this intelligence is default (implicit in the system), but AHAM recognizes that an author can explicitly provide a *teaching model* consisting of *pedagogical rules*.

The key elements in AHAM are thus the *domain model* (DM), *user model* (UM) and *teaching model* (TM). This division of adaptive hypermedia applications provides a clear separation of concerns when developing an adaptive hypermedia application. The main shortcoming in many current AHS is that these three factors or components are not clearly separated:

- The relationship between pages and concepts is too vague (e.g. [PDS98]) or too strict (e.g. [DC98]).
- The pedagogical rules can not be defined at the conceptual level but only at the page level (e.g. [DC98], [BSW96a], [BSW96b]).
- There is a mismatch between the high level of detail in the user model and the low reliability of the information on which an AHS must update that user model: for example, do access times represent reading times?

In this paper, we concentrate on the user modeling aspects of AHAM, but also describe how these aspects relate to the domain model and the teaching model. The rest of the paper is organized as follows. In Section 2 we introduce AHAM as a reference model for the design of hypermedia applications. A more detailed description can be obtained in [DHW99]. Note that some information on authoring support for adaptive hypermedia applications (in the context of AHAM) is available in [WHD99]. In Section 3 we focus on the aspect of user modeling and its influence on the adaptation. We show how user features are represented using attribute/value pairs in the context of AHAM. We describe the role of *knowledge values* in the representation of the user's knowledge on concepts from the domain model. We also illustrate how the actual adaptation process is based on the user model and how the maintenance of the user model can be facilitated. As a concrete example of the use of AHAM, Section 4 addresses the specific details of user modelling and adaptation in the AHA system. The way in which the domain model, user model and teaching model are implemented in this specific system are described. Note that this has led to a better insight in the consequences of some original design decisions, which resulted in a number of improvements to the original system. Using this approach we have thus been able to pinpoint some shortcomings in current adaptive hypermedia systems, not just in AHA. Section 5 concludes and indicates some future developments.

2. AHAM, a Dexter-based Reference Model

The most important aspects of hypermedia applications are the information nodes and the link structure connecting these nodes. In the Dexter reference model [HS90, HS94] this is captured in what Dexter calls the *Storage Layer*. This layer represents the application author's view on the application domain in terms of concepts. We call this view the *domain model* DM.

In adaptive hypermedia applications the central role of DM is shared with a *user model* UM. UM represents the relationship between the user and DM by keeping track of how much the user knows about each of the concepts in the application domain.

In order to perform adaptation (based on DM and UM) the author needs to specify how the user's knowledge influences the presentation of the information (from DM). In AHAM this is expressed by means of a *teaching model* TM consisting of pedagogical rules: the rules in TM model the explicit "intelligence" that the author wants the system to use in the adaptation process. An adaptive engine (as part of the AHS) then uses these rules to manipulate link anchors (from the Dexter

model's *Anchoring*) and to generate what the Dexter model calls the *Presentation Specifications*. Figure 1 shows the global structure of adaptive hypermedia applications in the AHAM model, just like Dexter focusing on the Storage Layer.

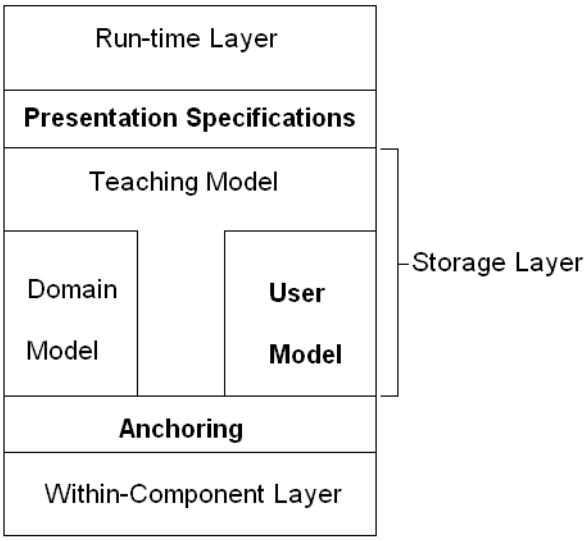

Figure 1: global structure of adaptive hypermedia applications.

2.1 The domain model

A *component* is an abstract notion in an AHS. It is a pair (uid, cinfo) where uid is a globally unique (object) identifier for the component and cinfo represents the component's information that consists of:

- a set of attribute-value pairs;
- a sequence of anchors (for attaching links);
- a presentation specification.

A *concept* is a component representing an abstract information item from the application domain. An *atomic concept* corresponds to a fragment of information: these are the primitive (non-adaptable) information units in the model with attribute and anchor values that belong to the Within-Component Layer. A *composite concept* has two "additional" attributes:

- a sequence of children (concepts);
- a constructor function (to denote how the children belong together).

There are a number of constraints. For example, the children of a composite concept are all atomic concepts (then it is a *page* (sequence of fragments) or in typical hypertext terminology a *node*) or they are all composite concepts. Figure 2 illustrates a part of a concept hierarchy.

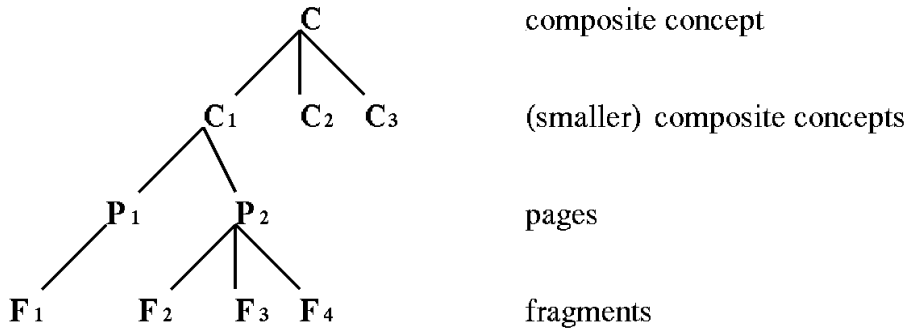

Figure 2: Example concept hierarchy.

An *anchor* is a pair (aid, avalue), where aid is a unique identifier for the anchor within the scope of its component and avalue is an arbitrary value that specifies some "location" within a concept component. Anchor values of atomic concepts belong to the Within-Component Layer, while anchor values of composite concepts are identifiers of concepts that belong to that composite.

A *specifier* is a tuple (uid, aid, dir, pres), where uid is the identifier of a concept, aid is the identifier of an anchor, dir is a direction (FROM, TO, BIDIRECT, or NONE), and pres is a presentation specification.

A *concept relationship* is a component, with two additional attributes:

- a sequence of specifiers;
- a concept relationship type.

The most common type of concept relationship is the type **link**, which corresponds to links in most hypermedia systems (with typically at least one FROM element and one TO or BIDIRECT element). In AHAM we consider other types of relationships as well, which play a role in the adaptation, such as for example relationships of the type **prerequisite**. When C_1 is a prerequisite for C_2 it means that the user should read C_1 before C_2. It does not mean that there must be a link from C_1 to C_2. It only means that the system somehow takes into account that reading about C_2 is not desired before some (enough) knowledge about C_1 has been acquired.

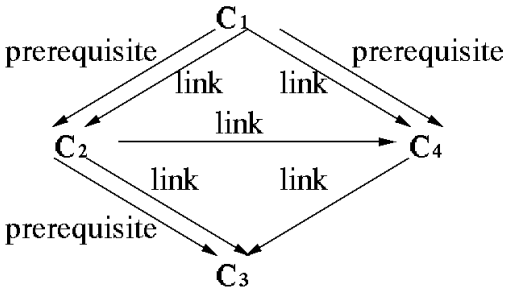

Figure 3: Example concept relationship structure.

Together, the atomic concepts, composite concepts and concept relationships build the *domain model* DM of an application.

2.2 The user model

An AHS associates a number of *user model attributes* with each concept component of DM. For every user the AHS maintains a *table-like structure*, in which for each concept the (user-specific) attribute values for that concept are stored. Note that these attribute values do not necessarily just represent the *knowledge level* of a concept. In Section 3 we look at the user model in more detail.

Note that since the user model consists of "named entities" for which we store a number of attribute/value pairs, there is no reason to limit these "entities" to pure *concepts* and their *knowledge level*. These concepts can be used to represent other user features, such as preferences, goals, background and hyperspace experience. For the AHS (and AHAM) the actual meaning of concepts is irrelevant.

2.3 The teaching model

The adaptation of the information content and of the link structure is based on a set of *rules*. In an application these rules build the connection between DM, UM and the presentation (specification) to be generated. Consider the following example:

- In DM one expresses that a concept C_1 is a prerequisite for concept C_2, and that there is a hypertext link from C_1 to C_2.
- In UM one expresses how much the user knows about concept C_1 (and C_2).

A rule is then needed to determine how "relevant" or "desirable" concept C_2 is, depending on the knowledge level about C_1. Other rules then express the effect of "relevance" on *adaptive navigation* (AN) and *adaptive presentation* (AP):

- (AN) One rule expresses how the relevance of C_2 influences the presentation of links in graphical maps.
- (AN) Another rule expresses how the relevance of C_2 influences the presentation of link anchors for links leading to C_2.
- (AP) The relevance of C_2 may influence the presentation of C_2 itself. For instance, an undesirable fragment may be hidden or grayed out.
- (AP) The relevance of elements of a composite may influence their selection or order. Fragments on a page may be sorted from most relevant to least relevant.

A *generic pedagogical rule* is a tuple (R, CRT, PH, PR), where R is a "triggered" rule, CRT is an optional concept relationship type, PH is the "phase" for the execution of the rule (either **pre** or **post**) and PR is a Boolean "propagate" field which indicates whether this rule may trigger other rules (to avoid infinite loops). The phase **pre** is executed during the generation of the page, while **post** is executed afterwards and is used for updating the user model. If a generic rule contains more than one concept it must have a CRT and then applies to all concept relationships of type CRT. Rules about just one concept do not have a CRT.

The syntax of the permissible rules depends on the AHS. Generic pedagogical rules are often system-defined, meaning that an author does not need to specify them. Author-defined rules always take precedence over (conflicting) system-defined rules. (Note that some AHS do not provide the possibility for authors to define their own generic pedagogical rules.)

A *specific pedagogical rule* is a tuple (R, PH, PR), where rule R uses (concrete) concepts from DM. (PH and PR are as for generic rules.) While specific rules are typically used to create exceptions to generic rules they can also be used to perform some ad-hoc adaptation based on concepts for which DM does not provide a relationship. Specific pedagogical rules must always be defined by the author. They take precedence over generic rules that would apply to the same concepts.

The *teaching model* TM of an AHS is the set of (generic and specific) pedagogical rules.

An AHS does not only have a DM, UM and TM, but also an *adaptive engine* AE. The adaptive engine provides the implementation dependent aspects while DM, UM and TM describe the information and adaptation at the conceptual, implementation independent level. It is a software environment that performs the following functions:

- It offers generic page selectors and constructors (used to determine which page to display when the user follows a link to a composite concept, or how to build a presentation for a page).
- It optionally offers a (very simple programming) language for describing new page selectors and -constructors.
- It performs adaptation by executing the page selectors and constructors (e.g. selecting a page, selecting fragments, manipulating link anchors).
- It updates the user model (instance) each time the user visits a page (by triggering the necessary pedagogical rules for the **post** phase).

An *adaptive hypermedia application* is a 4-tuple (DM, UM, TM, AE), where DM is a domain

model, UM is a user model, TM is a teaching model, and AE is an adaptive engine.

3. User Modeling and Adaptation in AHAM

3.1 Representation of user features using (attribute/value) pairs

By definition adaptive hypermedia applications reflect some features of the user in the user model. This model expresses various visible aspects of the system that depend on the user and that are visible to that user. Generally, there are five user features that are used by existing AHS [B96]:

- knowledge,
- user goals,
- background,
- hyperspace experience,
- preferences.

Almost every adaptive presentation technique relies on the user's *knowledge* as a source of adaptation. The system has to recognize the changes in the user's knowledge state and update its user model accordingly. The user's knowledge is often represented by an overlay model (overlay over the concepts from DM) or a (simpler) stereotype user model. As many adaptation techniques require a rather fine-grained approach, stereotype models are often too simple to provide adequate personalization and adaptation. Overlay models on the other hand are generally hard to initialize. Acceptable results are often achieved by combining the two kinds of modeling: starting with stereotype modeling and then moving towards a more fine-grained overlay model. Using the AHAM definition for user model, it is fairly straightforward how a user's knowledge state can be represented by associating a *knowledge value* attribute to each concept.

Apart from the concept's identifier (which may be just a name) a typical AHS will store not only a *knowledge value* for each concept, but also a *read* value which indicates whether (and how much) information about the concept has been read by the user, and possibly some other attribute values as well. While the model uses a table representation, implementations of AHS may use different data structures. For instance, a logfile can be used for the *read* attribute. The table below illustrates the (conceptual) structure of a user model for a course on hypermedia.

concept name (uid)	Knowledge value	read	...
Xanadu	well learned	true	...
KMS	Learned	true	...
WWW-page1	well learned	true	...
WWW-page2	not known	false	...
WWW	Learned	false	...
...	...	...	...

Table 1: Example user model (instance).

Note that AHAM's user model UM has enough expressive power to model all user features that current AHS take into account: the knowledge value of a concept can be a Boolean, discrete or continuous value depending on the choice of the author.

The second kind of user feature is the user's *goal*. The user's goal or task is a feature that is related with the context of the user's working activities rather than with the user as an individual. It is the most volatile of all user features. One possible representation of the user's current goal uses a set of pairs (Goal,Value), where Value is the probability that Goal is the current goal of the user. This representation perfectly matches the way in which AHAM models the user's state.

Two features of the user that are similar to the user's knowledge of the subject but that functionally differ from it, are the user's *background* and the user's *experience* in the given hyperspace. By background we mean all the information related to the user's previous experience *outside* the subject of the hypermedia system. By user's experience in the given hyperspace we mean how familiar is the user with the structure of the hyperspace and how easy can the user navigate in it. Again, these features can be modeled in AHAM using concepts' attribute/value pairs.

For different possible reasons the user can prefer some nodes and links over others or some parts of a page over others. This is used most heavily in information retrieval hypermedia applications. In fact in most adaptive information retrieval hypermedia applications *preferences* are the only information that is stored about the user. These user features differ from other ones, since in most cases they cannot be deduced by the system: the user has to inform the system directly or indirectly about the preferences. AHAM's concepts' attribute/value pairs can again be used to model the user's preferences.

From the above descriptions we can conclude that although a user model needs to represent (five) very different aspects of a user, all of these kinds of aspects can be implemented as sets of *concepts with associated attribute/value pairs*.

3.2 Adaptation based on the user model

The adaptive engine realizes adaptive presentation and adaptive navigation according to the pedagogical rules that are either system-defined or written by the author and that depend on the user model. Below we give a number of examples to show how pedagogical rules are used to do adaptation. The syntax used for the rules is arbitrary and only exemplary: AHAM does not prescribe any specific syntax. Normally every AHS will provide its own syntax for defining pedagogical rules.

Example 1 *For atomic concepts (fragments) let us assume that the presentation specification is a two-valued (almost Boolean) field, which is either "show" or "hide". When a page is being accessed, the following rule sets the visibility for fragments that belong to the page, depending on their "relevance" attribute-value.*

< access(C) and F IN C.children and F.relevance = true => F.pres := show, pre, false >

Here we simplified things, by assuming that we can treat C.children as if it were a set, whereas it really is a sequence. It is common to execute rules for generating presentation specifications in the **pre** phase, as done in this example.

Example 2 *The following rules set the presentation specification for a specifier that denotes a link (source) anchor depending on whether the destination of the link is considered relevant and whether the destination has been read before. For simplicity we consider a link with just one source and one destination.*

< CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.relevant = true and CR.ss[2].uid.read = false => CR.ss[1].pres = GOOD, pre, false >

< CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.relevant = true and CR.ss[2].uid.read = true => CR.ss[1].pres = NEUTRAL, pre, false >

< CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.relevant = false => CR.ss[1].pres = BAD, pre, false>

These rules say that links to previously unread but "relevant" pages are "GOOD". Links to previously read and "relevant" pages are "NEUTRAL" and links to pages that are not "relevant" are "BAD". In the AHA system [DC98] this results in the link anchors being colored blue, purple or black respectively (unless the user changes these preferences), while in ELM-ART [BSW96a] and Interbook [BSW96b] the links would be annotated with a green, yellow or red ball.

3.3 Maintenance of user model

To record the reading history of the user and the evolution of the user's knowledge, the system updates the user model based on the observation of the user's browsing process. The rules that the author has defined in TM describe how to keep track of the evolution of the user's knowledge.

Example 3 *The following rule expresses that when a page is accessed the "read" user-model attribute for the corresponding concept is set to true in the post phase:*

< access(C) => C.read := true, post, true >

This rule also says (through PR being true) that it will trigger other rules that have *read* on their left-hand side.

Example 4 *The following rule expresses that when a page is "relevant" and it is accessed, the knowledge value of the corresponding concept becomes "well-learnt" in the pre phrase. This is somewhat like the behavior of Interbook [BSW96b].*

< access(C) and C.relevant = true => C.knowledge := well-learnt, pre, true>

In Interbook, as well as in AHA [DC98], knowledge is actually updated in the **pre** phase. At the end of Section 4 we shall describe why this option is chosen, and which problems it creates. In general one wishes to have the option to base some adaptation on the knowledge state *before* accessing a page and some adaptation on the knowledge state *after* reading the page.

4. User Modeling and Adaptation in the AHA system

AHA [DC98] is a simple adaptive hypermedia system. We first describe the properties of the version that is currently being used for two on-line courses and one on-line information kiosk. At the end of this section we indicate planned changes for the next version of AHA.

- In AHA the *domain model* consists of three types of concepts: *abstract concepts*, *fragments* and *pages*. Concepts are loosely associated with (HTML) pages, not with fragments.
- The *user model* consists of:
  - Color preferences for link anchors which the user can customize. (These preferences result in "non-relevant" link anchors to be hidden if their color is set to black, or visibly "annotated" if their color is set to a non-black color, different from that of "relevant" link anchors.)
  - For each abstract concept, a Boolean *knowledge* attribute. (*True* means the concept is known, *false* means it is not known.)

- For each page, a Boolean *read* attribute. (*True* means the page was read, *false* means it was not read.) AHA actually logs access and reading times, but they cannot be used in a more sophisticated way in the current version.

- AHA comes with a *teaching model* containing system-defined generic pedagogical rules. It offers a simple language for creating author-defined specific pedagogical rules (but no author-defined generic rules).

The *domain model* can only contain concept relationships of the types that are shown below. The influence of these relationships on the adaptation and the user model updates is defined by system-defined generic pedagogical rules. In AHA all rules are executed in the **pre** phase and are triggered directly by a page access, thus eliminating the need for propagation (causing problems that we describe later).

- When a page is accessed, its *read* attribute in the user model is updated:

  < access(P) => P.read := true, pre, false >

- The relationship type *generates* links a page to an abstract concept. A *generates* relationship between P and C means that reading page P generates knowledge about C:

  < access(P) => C.knowledge := true, pre, false >

  This "generation" of knowledge in AHA is controlled by a structured comment in an HTML page:

  ```
  <!-- generates readme -->
  ```

  This example *generates* comment denotes that the concept `readme` becomes known when the page is accessed.

- The relationship type *requires* links a page to a composite concept which is defined by a Boolean expression of concepts. Although in principle this composite concept is unnamed, we shall use a "predicate" or "pseudo attribute of the page" to refer to it: *P.requires* is used as a Boolean attribute of which the value is always that of the corresponding Boolean expression. A *requires* relationship is implemented using a structured comment at the top of an HTML page, e.g.:

  ```
  <!-- requires ( readme and intro ) -->
  ```

  This example expresses that this page is only considered relevant when the concepts `readme` and `intro` are both known. In AHA, links to a page for which *requires* is *false* are considered BAD, and reading such a page does not generate knowledge. Below we give the rules that determine how the link anchors will be presented. They are very similar to the rules in Example 2 (Subsection 3.2):

  < CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.requires = true and CR.ss[2].uid.read = false => CR.ss[1].pres = GOOD, pre, false >

  < CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.requires = true and CR.ss[2].uid.read = true => CR.ss[1].pres = NEUTRAL, pre, false >

  < CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.requires = false => CR.ss[1].pres = BAD, pre, false>

- The relationship type *link* only applies to pairs of pages in AHA.

In AHA structured HTML comments are used for specifying author-defined specific pedagogical rules about the conditional inclusion of fragments in HTML pages. With a fragment F we can associate a "pseudo attribute" *requires* to indicate the condition, just like for whole pages. The syntax is illustrated by the following example:

```
<!-- if ( readme and not intro ) -->
... here comes the content of the fragment ...
<!-- else -->
... here is an alternative fragment ...
<!-- endif -->
```

AHA only includes fragments when their *requires* "attribute" is *true*.

We conclude this section with an illustration of one specific shortcoming which we have found in both AHA [DC98] and Interbook [BSW96b]: the "new" knowledge values are calculated in the **pre** phase (and in fact these systems do not support a two-phase approach at all). When a user requests a page, the knowledge generated by reading this page is already taken into account during the generation of the page. This has desirable as well as undesirable side-effects:

- When links to other pages become relevant *after* reading the current page it makes sense to already annotate the link anchors as relevant when presenting the page. Once a page is generated its presentation remains static while the user is reading it (and rightfully so). The new knowledge thus needs to be taken into account before the page is actually read.

- Pages contain information that becomes relevant or non-relevant depending on the user's knowledge. In some cases the relevance of a fragment may depend on the user having read the page that contains this fragment. This means that a fragment may be relevant the first time a page is visited and non-relevant thereafter, or just the other way round. By already taking into account the knowledge before the page is generated for the first time a different "first time version" becomes impossible to create.

5. Conclusion

We have developed a reference model for adaptive hypermedia applications, named AHAM. The description of adaptive applications in terms of AHAM has provided us with valuable redesign issues for the AHS AHA. The two most important ones are:

- The division of an adaptive hypermedia application into a *domain model*, *user model*, and *teaching model* provides a clear separation of concerns and will lead to a better separation of orthogonal parts of the AHS functionality in the implementation of the next version of AHA. We believe that any system which supports this separation of concerns will not only result in a cleaner implementation, but also in a more usable authoring environment.
- By representing AHA in the AHAM model we have identified another shortcoming: the lack of a two-phase application of rules. We found that this shortcoming is present in other AHS as well.

In this paper we have focused on the user modeling aspects of AHAM, and have described how they relate to the domain model and the teaching model. We have shown how user features are modeled and how the actual adaptation process is performed. We have illustrated this general approach for user modeling and adaptation by explicitly considering the AHS AHA. The description of these user model aspects at an abstract level sets AHAM apart from other descriptions of AHS which are too closely related to the actual implementation of these AHS.

References

[B96] Brusilovsky, P., “Methods and Techniques of Adaptive Hypermedia”. User Modeling and User-Adapted Interaction, 6, pp. 87-129, 1996. (Reprinted in Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers, pp. 1-43, 1998.)

[CD97] Calvi, L., De Bra, P., “Using Dynamic Hypertext to create Multi-Purpose Textbooks”. Proceedings of ED-MEDIA’97, Calgary, pp. 130-135, 1997.

[DC97] De Bra, P., Calvi, L., “Creating adaptive hyperdocuments for and on the Web”. Proceedings of the WebNet’97 Conference, Toronto, pp. 149-165, 1997.

[DC98] De Bra, P., Calvi, L., “AHA: a Generic Adaptive Hypermedia System”. Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia, Pittsburgh, pp. 5-11, 1998.

[DHW99] De Bra, P., Houben, G.J., Wu, H., “AHAM: A Dexter-based Reference Model for Adaptive Hypermedia”. Proceedings of ACM Hypertext’99, Darmstadt, pp. 147-156, 1999.

[HS90] Halasz, F., Schwartz, M., “The Dexter Reference Model”. Proceedings of the NIST Hypertext Standardization Workshop, pp. 95-133, 1990.

[HS94] Halasz, F., Schwartz, M., “The Dexter Hypertext Reference Model”. Communications of the ACM, Vol. 37, nr. 2, pp. 30-39, 1994.

[PDS99] Pilar da Silva, D., “Concepts and documents for adaptive educational hypermedia: a model and a prototype”, Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia, Pittsburgh, pp. 33-40, 1998.

[WHD99] Wu, H., Houben, G.J., De Bra, P., “Authoring Support for Adaptive Hypermedia”, Proceedings ED-MEDIA’99, Seattle, pp. 364-369, 1999.